

### **REMARKS**

Claims 1, 2, 4-13, 15-24, and 26-39 are pending. Claims 1, 5, 13, 16, 24, 27, 30, 35, and 36 have been amended. New claims 37-39 have been added. No new matter has been introduced. Reexamination and reconsideration of this application is respectfully requested.

In the January 26, 2005 Office Action, the Examiner objected to the drawings for being informal. Attached herewith are formal drawings. Therefore, applicants respectfully submit that this objection is obviated.

The Examiner rejected claims 1, 2, 4-13, 15-24, and 26-36 under §112, ¶1. The Examiner rejected claims 1, 2, 4-13, 15-24, and 26-36 under 35 U.S.C. §103(a) as being obvious over the book “Visual Modeling with Rational Rose 2000 and UML,” ISBN: 0-201-69961-3, by Quatrani (“Quatrani”) in view of the online brochure “Accelerating Embedded e-development” located at <[www.ghs.com/partners/rational/rose-rt.pdf](http://www.ghs.com/partners/rational/rose-rt.pdf)> (“Accelerating”). These rejections are respectfully traversed.

### **§112, ¶1 rejection**

Claims 1, 2, 4-13, 15-24, and 26-36 were rejected under §112, ¶1. Specifically, the Examiner stated that there was no support in the original disclosure for the limitation “generating software code ... in real time.” Applicants respectfully disagree with the Examiner’s assessment and note that support for this limitation is contained at, *e.g.*, paragraph 1, lines 1-4, which recites (with emphasis added): “[t]he present invention relates generally to *real-time embedded software* for a game of chance and, more particularly, to a method and apparatus for *automatically generating such software* from formal design models.” Additional support is contained at, *e.g.*, paragraph 6, lines 3-5; paragraph 24, lines 1-3; the preambles of claims 1 and 13 as originally filed; and the abstract. Accordingly, applicants respectfully submit that the

rejection of claims 1, 2, 4-13, 15-24, and 26-36 were rejected under §112, ¶1 should be withdrawn.

The Quatrani and Accelerating references

Quatrani discloses visual modeling within the Rational Rose family of products and use of Universal Markup Language (“UML”). Quatrani also discloses the Rational Unified Process, which is an extensive set of guidelines that address the technical and organizational aspects of software development focusing on requirements analysis and design. The Examiner stated that Quatrani further discloses “preparing an analysis model, ... preparing a design model, ... [and] generating software code for the application from the design model, the software code including at least a portion that is automatically generated using a software development tool”. [Jan. 26, 2005 Office Action, Pp. 3-4.]

The Examiner stated that the automatic generation of software code is disclosed on page 13, where Quatrani sets forth: “Structuring the project along the process component dimension includes the following activities: ... Implementation – the production of the code that will result in an executable system.” [*Id.* at p. 6.] With respect to design models, the Examiner stated that Quatrani discloses a design manual that defines static relationships and dynamic behavior of objects. For this proposition, the Examiner referred to a passage in Quatrani, which recites: “[u]se cases and scenarios provide a way to describe system behavior; that is, the interaction between objects in the system... A statechart diagram shows the states of a single object, the events or messages that cause a transition from one state to another, and the actions that result from a state change.” [Quatrani, P. 131.<sup>1</sup>]

---

<sup>1</sup> Applicants note that the Examiner did not provide a copy of the Quatrani reference. Instead, the Examiner provided only an Internet link to a website to which the applicants are not registered. Accordingly, applicants purchased a copy of the Quatrani reference. Applicants provide, in appendix A, a copy of all pages of Quatrani

The Examiner noted that Quatrani *does not disclose* a method for generating real-time embedded software code for an application. However, the Examiner stated that Accelerating does disclose such real-time method. Accelerating discloses the following on page 1: “[Rational Rose RealTime] enhances the power of industry standard UML with a real-time profile optimized for the unique problems of concurrency and distribution.... A UML model compiler generates complete C and C++ applications for UNIX, Windows NT/2000, and real-time operating system targets. This automated code generation eliminates the need for manual translation and avoids costly design interpretation errors.” The Examiner also stated that it would have been obvious to a person of skill in the art at the time the invention was made to combine the references in the direction of the claims.

Claims 1, 2, 4-13, 15-24, and 26-34

Claims 1, 2, 4-13, 15-24, and 26-34 all contain, either directly or indirectly (via claim dependencies), at least the distinguishing limitations discussed below. Representative independent claim 1 recites (with emphasis added):

1. A method for generating real-time embedded software code **for a wagering game having a randomly-selected outcome**, comprising:
  - preparing an analysis model **for the wagering game**, the analysis model describing functionality to be included in the software code;
  - preparing a design model **for the wagering game**, the design model including a plurality of objects for realizing the functionality in the analysis model, wherein the design model defines **static relationships** between the objects and dynamic behavior of the objects, **wherein the functionality realized is determined by the design model based on the analysis model**; and
  - generating software code **for the wagering game** from the design model, the software code including at least a portion that is automatically generated in real-time using a software development tool, wherein the automatically generated portion of the software code includes the static relationships between the objects and the dynamic behavior of the objects.

---

referenced in this Amendment in case the page numbers cited differ from those in the Examiner's copy of the Quatrani.

Neither Quatrani nor Accelerating, alone or in combination, disclose, teach, or suggest a method for generating real-time embedded software code for a *wagering game having a randomly-selected outcome*. Instead, Quatrani is directed to Rational Rose 2000 visual modeling systems for complex systems. [See Quatrani, P. 3] Accelerating is directed to an extension of Rational Rose that provides UML model code generation and visualization. However, there is no suggestion anywhere in either of these references that it would have been obvious to apply their teachings to a *wagering game having a randomly-selected outcome*.

Moreover, claim 1 further specifies preparing a design model for the wagering game, the design model including a plurality of objects for realizing the functionality in the analysis model, where the *design model defines static relationships between the objects* and dynamic behavior of the objects. Quatrani does not disclose preparing a design model that defines static relationships between objects and dynamic behavior of the objects. The Examiner stated that Quatrani discloses that the design model defines static relationships, and cited the following passage from page 131 of Quatrani (emphasis in original):

**“MODELING DYNAMIC BEHAVIOR**

USE CASES AND scenarios provide a way to describe system behavior; that is, the interaction between objects in the system. Sometimes it is necessary to look at the behavior inside an object. A statechart diagram shows the states of a single object, the events or messages that cause a transition from one state to another, and the actions that result from a state change.”

Applicants disagree with the Examiner’s assessment that the above paragraph discloses use of static relationships. Instead, as the headline indicates, this paragraph describes only the *dynamic behavior* of objects in the static, not the *static relationships* between the objects.

Claim 1 further requires that *the functionality realized is determined by the design model based on the analysis model*. This limitation is not taught by Quatrani. Specifically, on page 9, Quatrani discloses that the structuring of a project includes a number of activities, including

“Analysis and Design – a description of how the system will be realized in the implementation phase.” Applicants note that this disclosure in Quatrani is very general and vague and certainly does not teach, suggest, or provide an enabling disclosure that separate design and analysis models are utilized, where the functionality realized is determined by the *design model based on the analysis model*.

Accordingly, for at least these reasons, claims 1, 2, 4-13, 15-24, and 26-34 all distinguish over Quatrani, alone or in combination with Accelerating.

Claim 5, 6, 16, 17, 27, and 28 further distinguish over Quatrani and Accelerating. Specifically, representative claim 5 recites (with emphasis added): “wherein the functionality described by the analysis model is organized into use cases, **the use cases being selected from the group consisting of: handling money, playing the wagering game, handling critical events, and servicing the machine.**” Quatrani discloses using use cases that model a dialogue between an actor and the system. However, Quatrani does not, alone or in combination with Accelerating, disclose, teach, or suggest that the use cases are handling money, playing the wagering game, handling critical events, or servicing the machine. Therefore, claims 5, 6, 16, 17, 27, and 28 further distinguish over Quatrani.

Claim 30 further distinguishes over Quatrani, alone or in combination with Accelerating. Specifically, claim 30 recites (with emphasis added): “[t]he collection of claim 24, wherein the **wagering game is a slot reel game including a plurality of symbol-bearing reels that are rotated and stopped to place symbols on the reels in visual association with a display area.**” As discussed above, neither Quatrani nor Accelerating are directed to a wagering game. Accordingly, it follows that neither of Quatrani nor Accelerating, alone or in combination, disclose, teach, or suggest that the wagering game is a slot reel game including a plurality of

symbol-bearing reels that are rotated and stopped to place symbols on the reels in visual association with a display area. Therefore, claim 30 further distinguishes over Quatrani, alone or in combination with Accelerating.

Accordingly, applicants respectfully submit that claims 1, 2, 4-13, 15-24, and 26-34 distinguish over Quatrani, alone or in combination with Accelerating and the rejection of these claims under 35 U.S.C. §103(a) should be withdrawn.

### Claim 35

Claim 35 distinguishes over Quatrani, alone or in combination with Accelerating. Claim 35 recites (with emphasis added):

35. A method for generating real-time embedded software code for a **wagering game having a randomly-selected outcome**, comprising:

preparing an analysis model for the **wagering game**, the analysis model describing functionality to be included in the software code, **wherein the functionality described by the analysis model is organized into use cases, the use cases being selected from the group consisting of: handling money, playing the wagering game, handling critical events, and servicing the machine;**

preparing a design model for the **wagering game**, the design model including a plurality of objects for realizing the functionality in the analysis model;

generating software code for the **wagering game** from the design model, the software code including at least a portion that is automatically generated in real-time using a software development tool; and

modifying the design model and automatically modifying the software code in response to modifying the design model.

Claim 35 contains distinguishing limitations similar to those discussed above with respect to claim 5. Specifically, claim 35 specifies that the analysis model includes use case diagrams, where *the use cases are selected from the group consisting of: handling money, playing the wagering game, handling critical events, and servicing the machine*. Such use case diagrams are neither disclosed, taught, nor suggested by Quatrani, alone or in combination with Accelerating. Accordingly, claim 35 distinguishes over Quatrani, alone or in combination with Accelerating,

and applicants respectfully submit that the rejection of claim 35 under 35 U.S.C. §103(a) should be withdrawn.

Claim 36

Claim 36 also distinguishes over Quatrani, alone or in combination with Accelerating.

Claim 36 recites (with emphasis added):

36. A method for generating real-time embedded software code for a wagering game having a randomly-selected outcome, comprising:  
preparing an analysis model for the wagering game, the analysis model describing functionality to be included in the software code, **wherein the analysis model includes use case diagrams, the use case diagrams defining relationships between the use cases and external actors outside the wagering game, and the external actors are selected from the group consisting of: a player, a money handling function, a host, and a random number generator;**  
preparing a design model for the wagering game, the design model including a plurality of objects for realizing the functionality in the analysis model;  
generating software code for the wagering game from the design model, the software code including at least a portion that is automatically generated in real-time using a software development tool; and  
modifying the software code and automatically modifying the design model in response to modifying the software code.

Claim 36 specifies that the analysis model includes use case diagrams, the use case diagrams defining relationships between the use cases and external actors outside the wagering game, and *the external actors are selected from the group consisting of: a player, a money handling function, a host, and a random number generator*. Such external actors are neither disclosed, taught, nor suggested by Quatrani, alone or in combination with Accelerating. Accordingly, claim 36 distinguishes over Quatrani, alone or in combination with Accelerating, and applicants respectfully submit that the rejection of claim 36 under 35 U.S.C. §103(a) should be withdrawn.

Claims 37-39

New claims 37-39 depend from claims 1, 13, and 24, respectfully, and distinguish over Quatrani, alone or in combination with Accelerating for at least the same reasons as discussed above with respect to claims 1, 13, and 24. Moreover, claims 37-39 each recite an additional distinguishing limitation directed to use of external actors similar to the limitation discussed above with respect to claim 36. For example, representative claim 37 recites that **“the external actors are selected from the group consisting of: a player, a money handling function, a host, and a random number generator.”** (Emphasis added.) As discussed above, Quatrani does not disclose, teach, or suggest that the external actors are a player, a money handling function, a host, or a random number generator. Therefore, claims 37-39 further distinguish over Quatrani, alone or in combination with Accelerating.



Conclusion

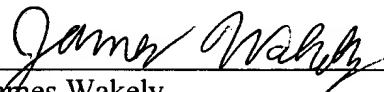
Applicants believe that the foregoing amendments place the application in condition for allowance, and a favorable action is respectfully requested. If for any reason the Examiner finds the application other than in condition for allowance, the Examiner is requested to call the undersigned attorney at the Chicago telephone number (312) 425-3900 to discuss the steps necessary for placing the application in condition for allowance should the Examiner believe that such a telephone conference would advance prosecution of the application.

A check in the amount of \$940.00 is enclosed for the fees due in connection with the present Request for Continued Examination and Amendment. The Commissioner is authorized to charge any required fees while this application is pending (except the issue fee) to Jenkins & Gilchrist, P.C. Deposit Account No. 10-0447(47079-00105USPT).

Respectfully submitted,

JENKENS & GILCHRIST, P.C.

Date: June 24, 2005

  
James Wakely  
Reg. No. 48,597  
ATTORNEY FOR APPLICANTS  
225 West Washington Street  
Suite 2600  
Chicago, IL 60606-3418  
(312) 425-3900  
(312) 425-3909 (fax)

## APPENDIX A

Best Available Copy

# VISUAL MODELING WITH RATIONAL ROSE 2000 AND UML

TERRY QUATRANI



ADDISON-WESLEY

Boston • San Francisco • New York • Toronto • Montreal  
London • Munich • Paris • Madrid  
Capetown • Sidney • Tokyo • Singapore • Mexico City

## WHAT IS VISUAL MODELING?

VISUAL MODELING IS a way of thinking about problems using models organized around real-world ideas. Models are useful for understanding problems, communicating with everyone involved with the project (customers, domain experts, analysts, designers, etc.), modeling enterprises, preparing documentation, and designing programs and databases. Modeling promotes better understanding of requirements, cleaner designs, and more maintainable systems.

Models are abstractions that portray the essentials of a complex problem or structure by filtering out nonessential details, thus making the problem easier to understand. Abstraction is a fundamental human capability that permits us to deal with complexity. Engineers, artists, and craftsmen have built models for thousands of years to try out designs before executing them. Development of software systems should be no exception. To build complex systems, the developer must abstract different views of the system, build models using precise notations, verify that the models satisfy the requirements of the system, and gradually add detail to transform the models into an implementation.

We build models of complex systems because we cannot comprehend such systems in their entirety. There are limits to the human capacity to understand complexity. This concept may be seen in the world of architecture. If you want to build a shed in your backyard, you can just start building; if you want to build a new house, you probably need a blueprint; if you are building a skyscraper, you definitely need a blueprint. The same is true in the world of software. Staring at lines of source code or even analyzing forms in Visual Basic does little to provide the programmer with a global view of a development project. Constructing a model allows the designer to focus on the big picture of how a project's components interact, without having to get bogged down in the specific details of each component.

Increasing complexity, resulting from a highly competitive and ever-changing business environment, offers unique challenges to system developers. Models help us organize, visualize, understand,

Structuring a project along the time dimension involves the adoption of the following time-based phases:

- Inception—specifying the project vision
- Elaboration—planning the necessary activities and required resources; specifying the features and designing the architecture
- Construction—building the product as a series of incremental iterations
- Transition—supplying the product to the user community (manufacturing, delivering, and training)

Structuring the project along the process component dimension includes the following activities:

- Business Modeling—the identification of desired system capabilities and user needs
- Requirements—a narration of the system vision along with a set of functional and nonfunctional requirements
- Analysis and Design—a description of how the system will be realized in the implementation phase
- Implementation—the production of the code that will result in an executable system
- Test—the verification of the entire system
- Deployment—the delivery of the system and user training to the customer

Figure 1-4 shows how the process components are applied to each time-based phase.

Each activity of the process component dimension typically is applied to each phase of the time-based dimension. However, the degree to which a particular process component is applied is dependent upon the phase of development. For example, you may decide to do a proof of concept prototype during the Inception Phase, and thus, you will be doing more than just capturing requirements (you will be doing the analysis, design, implementation, and test needed to complete the prototype). The majority of the analysis process

## MODELING DYNAMIC BEHAVIOR

USE CASES AND scenarios provide a way to describe system behavior; that is, the interaction between objects in the system. Sometimes it is necessary to look at the behavior inside an object. A statechart diagram shows the states of a single object, the events or messages that cause a transition from one state to another, and the actions that result from a state change.

A statechart diagram will not be created for every class in the system, only for classes with "significant" dynamic behavior. Interaction diagrams can be studied to determine the dynamic objects in the system—ones receiving and sending many messages. Statechart diagrams are also useful to investigate the behavior of an aggregate "whole" class and of control classes.

Care must be taken to stay in an analysis frame of mind—concentrating on the WHAT of the problem and not the HOW of the solution.



### CREATING STATECHART DIAGRAMS IN RATIONAL ROSE

1. Right-click to select the class in the browser and make the shortcut menu visible.
2. Select the New:Statechart Diagram menu choice. This will add a state diagram called NewDiagram to the browser.
3. While the diagram is still selected, enter the name of the diagram.
4. To open the diagram, click the + to expand the class in the browser, click the + to expand the State/Activity Model in the browser and double-click on the statechartdiagram in the browser.

The browser view of the statechart diagram for the CourseOffering class is shown in Figure 9-1.